ITCS 209 Object Oriented Programming	Name:	**Lab Score**	**Challenge Bonus**	
	ID:			
	Section:			

Lab14: Sorting

You are to create a program of **WordProcessor.java** to process on the list of words. This class will read a text file, print out the words, sort the words (delimited by white space characters), and remove duplicate words. The first two methods i.e., `readFile` and `printWords` are provided (please see that starter code). Your task is to implement the remaining two methods:

1. The sort method: the ***static void*** method should accept a list of strings, then sort it in **descending** order (b comes before a) using one of the sorting algorithms discussed in class (i.e., Selection, Bubble, Insertion, or MergeSort). **After each pass, call the second method to print out the immediate result.** You must implement the sorting algorithm from scratch (i.e. You may not use Arrays.sort() and Collections.sort()). Test your program with **Main.java file** and be prepared to explain (in **English**) how your sorting algorithm works from the output.

Example output (Insertion Sort):

```
Original List:
[ink, data, ink, data, test, run, world, tech, mango]
-------------------------------------

Pass 1: [ink, data, ink, data, test, run, world, tech, mango]
Pass 2: [ink, ink, data, data, test, run, world, tech, mango]
Pass 3: [ink, ink, data, data, test, run, world, tech, mango]
Pass 4: [test, ink, ink, data, data, run, world, tech, mango]
Pass 5: [test, run, ink, ink, data, data, world, tech, mango]
Pass 6: [world, test, run, ink, ink, data, data, tech, mango]
Pass 7: [world, test, tech, run, ink, ink, data, data, mango]
Pass 8: [world, test, tech, run, mango, ink, ink, data, data]
```

Hint: str1.compareTo(str2) returns **0** if `str1` is equal to `str2`; a **negative value** if `str1` is lexicographically less than `str2`; and a **positive value** if `str1` is lexicographically greater than `str2`.

2. The removeDuplicate method: the ***static void*** method should accept a list of strings, then remove all duplicates from the list. For example, if the list has "ink sort make ink sort ink", then the list should be changed to "sort make ink". Note that the resulting list may not have the same ordering as the original one. One approach is to sort the list of words first. Then, for each element in the list, look at its next neighbor to decide whether it is present more than once. If so, remove it.

Example output :

```
Original List:
[ink, data, ink, data, test, run, world, tech, mango]
-------------------------------------

No Duplicate List:
[world, test, tech, run, mango, ink, data]
```

Note: The Movie.java & SoritngMovie.java will be used in the challenge section only.

Challenge Bonus (Optional):

Your task is to sort the movies in **<u>ascending</u>** order based on multiple criteria, the title ('a' comes before 'b'), released year (year 2012 comes before 2019), and movie ID (mid 2 comes before 3), respectively. If both movies have the same titles, their released years will be checked. If, again, their release years are equal, their movie ID will be compared.

```
== unsorted movie list ==
[mid:1 |The Intern |2009]
[mid:2 |The Gift |2009]
[mid:3 |The Lost Room |2009]
[mid:4 |The Gift |2012]
[mid:5 |Pasolini |2012]
[mid:6 |The Intern |2009]
[mid:7 |American Ultra |2019]
[mid:8 |Sweet Red Bean Paste |2019]
```

```
== sorted movie list (ascending) ==
[mid:7 |American Ultra |2019]
[mid:5 |Pasolini |2012]
[mid:8 |Sweet Red Bean Paste |2019]
[mid:2 |The Gift |2009]
[mid:4 |The Gift |2012]
[mid:1 |The Intern |2009]
[mid:6 |The Intern |2009]
[mid:3 |The Lost Room |2009]
```

As shown in the sorted list above, the movie mid 7 comes before mid 5 because "American Ultra" is lexicographically less than "Pasolini." The movie mid 2 and mid 4 have the same title, but mid 2 is released before mid 4, so mid 2 comes before mid 4. The movie mid 1 and mid 6 have the same title and released year, so we have to compare their mids. As a result, mid 1 comes before mid 6 because the value of 1 is smaller than the value of 6.

The `Movie` class implementing the `Comparable` interface is provided. You have to complete the unimplemented method **`int compareTo(Movie m).`**

For this method, `m1.compareTo(m2)` **returns 0** if m1 is equal to m2 (i.e., they have the same title, release year, and mid); **return a negative** value if m1 comes before m2, and **return a positive** value if m1 comes after m2.

Write a **sorting algorithm** in `SortingMovie.java` class and display the sorted movie list in ascending order.
